下载APP

07 | 系统设计能力考察：系统设计内功到底怎么考？

2021-04-20 四火

技术面试官识人手册 进入课程 >

讲述：四火
时长 17:23 大小 15.92M

你好，我是四火。

在上一讲，我们谈到对于数据结构和算法部分考察的把控，这一讲，我们来聊聊系统设计。

我们经常说软件工程师要注意积累工作经验，那么，年复一年的工作中，我们到底积累了哪些技术方面的经验？

我想，其中很重要的一个就是对于系统的理解。对于刚毕业和工作不久的工程师来说，我们不应苛求他们对系统具备相当深度的理解；另一方面，即便工作年数相同，不同候选人对于系统的认识深度也有着非常大的差距。

和能够明确判断出正误的算法类问题不同，系统设计我们往往没法给出丁是丁卯是卯的正误判断，因而面试中讨论的更多是可行性，以及对于不同实现方案的利弊权衡。这也是符合现实逻辑的，有一些相似的系统，它们的实现技术截然不同，有着不同的优势和短板，但是最终都解决了最核心的问题。

糟糕的例子

好，我们先来说几个典型的反面例子，和算法、数据结构的面试不同，系统设计的面试往往能遵循的简单套路更少，因而也能见到更多的“踩坑”实践。

排斥不一样的思路

最典型的例子就是面试官缺乏包容心。就像“看自己的代码越看越开心，看别人的代码越看越糟心”一样，在讨论一些经典系统设计思路的时候，如果对方的思路和常规思路明显不同，**某些面试官可能会立即觉得对方的想法很“奇葩”。**

这种想法其实是非常危险的，因为面试官在心里默默建立了一个护城墙，凡是不符合“标准”的解决思路，都会产生排斥心理。

下面我举一个具体例子说明吧。有一次，某位面试官和候选人讨论，怎样设计一个篮球比赛的文字直播系统。

这个系统需要实时显示当前的比分等信息，那么一种思路是使用普通的查询，客户端主动发起，即定时地不断询问服务端最新的比分；还有一种思路是使用长连接，或者 long poll 这样的机制，使得服务端可以更实时地推送最新的比分更新。

很多系统都采用后者的思路来实现，这主要是基于实时性等方面的考虑，但是当时候选人考虑使用前一种方案。于是，面试官就直接中断了他的陈述，并直接解释为什么应该选择后者。

暂且不说所谓的正误，或者好与不好。这个技术实现选择的不一致，本来可以引出一个很好的面试实践，也就是对两种实现机制的利弊做讨论，但很遗憾，这种打断的做法直接终止了这样的讨论。

也就是说，我们完全可以跟着候选人的思路，用上一讲提到的“要求澄清”的办法让他自己来分析出实现的利弊来。

之后，如果候选人自己意识到，或者面试官使用“给出挑战”这样的办法，通过讨论这个方案的不足，来引出后一个方案，并放到一起进行优劣比较。

这样一来就可以有比较多的讨论与互动，我们也能得到更多的关于候选人对于系统理解的数据点。我想强调的是，最后采用哪一个方案其实并不重要，重要的是这个过程和我们得到的考察数据。

还有种类似情况是，**有时候选人对于系统的设计思路是自底向上的，而面试官则是自上而下的。**

一般说来，自底向上去思考和设计系统的时候，确实存在“捡了芝麻丢了西瓜”的情况，但是如果面试中候选人这么做，我们不要去打断他，或者破坏他的这种思维模式，而要顺着他最熟悉的方法来讨论这个问题。

我们做系统设计面试的目的，是要考察候选人对于系统的理解，抓住一些有意义的数据点，而不是要尝试去把系统设计做到最好。

换言之，**我们在面试中，一定要摒弃掉所谓的“最佳设计”或者“推荐方案”这样的想法，而是尽量追着候选人的思路跑！**

最后设计出的系统也许有这样那样的问题，但请记住面试官可以主导面试，却最好不要主导设计——完成的系统一定是一个候选人主导设计的系统，而不是面试官。

无意义的深挖

我们当然可以针对系统的某一个点进行深挖，但是如果候选人已经暴露出，他还不具备深挖该问题的基础知识，那么这个继续深挖很可能就是徒劳的，因为它除了给候选人带来沮丧以外，并不能得到太多更有价值的考察数据点。

举个例子，有一位面试官和候选人一起讨论“短 URL 系统”的设计问题。其中一个点是，在做短 URL 重定向的时候，返回码应该选择 301 还是 302？

302 是临时重定向，301 是永久重定向，虽然 301 可能会减轻一定的访问压力，但通常来说，我们一般都选择 302。这里的原因是，这些系统有一个很大的价值就是可以获取用户的访问数据，使用 301 可能会使得用户后续的访问直接跳过了这个系统，而去访问重定向后的系统，这样，这些有价值的 URL 访问数据就丢失了。

但是，当时候选人只对 HTTP 返回码略有了解，并不知道 301 和 302 的意义是什么，回答也明显有些搪塞，这种情况下，面试官还想继续往下挖，问到底应该选择 301 还是 302。我觉得，这种深挖的意义就不大了。

考察技巧

好了，说完反面，下面来从正面讲讲系统设计考察的技巧。通常来说，系统设计的考察有两种主要形式：

第一种，是让候选人讲他自己熟悉的、做过的系统，或者是引以为豪的项目；

第二种，是像前几讲中介绍的那样，给出一个实际问题，细化、分析，延伸成一个对于核心功能的系统设计问题，并加以讨论和解决。

做过的系统

第一种面试方式非常常见。**让候选人谈论做过的系统，把握一条原则，是候选人必须非常熟悉，最好是“最”熟悉的系统**，这样我们才能了解到，候选人在过去的工作中，表现如何，有多深入的理解，又有多少思考。

我们不妨回想一下其它的技术方面的考察路径，如果是算法和数据结构，问“做过的算法”，好像很难这么问，而且这么问很可能也挖不到什么有价值的信息；如果是面向对象，问“做过的项目中，类和方法的设计”，听起来就觉得非常别扭，对不对？

确实如此，对于技术方面的考察点，唯有系统设计，问“做过的系统”，是一个非常好的途径。

话说回来，这样的面试方式，**也存在着一定的局限性。**

最大的问题，就是它只考察了候选人对于“已有系统”的理解，多为“经验之谈”，这是可以事先充分准备的。它并不能很好反映出，对于一个陌生的新问题，候选人能否灵活地

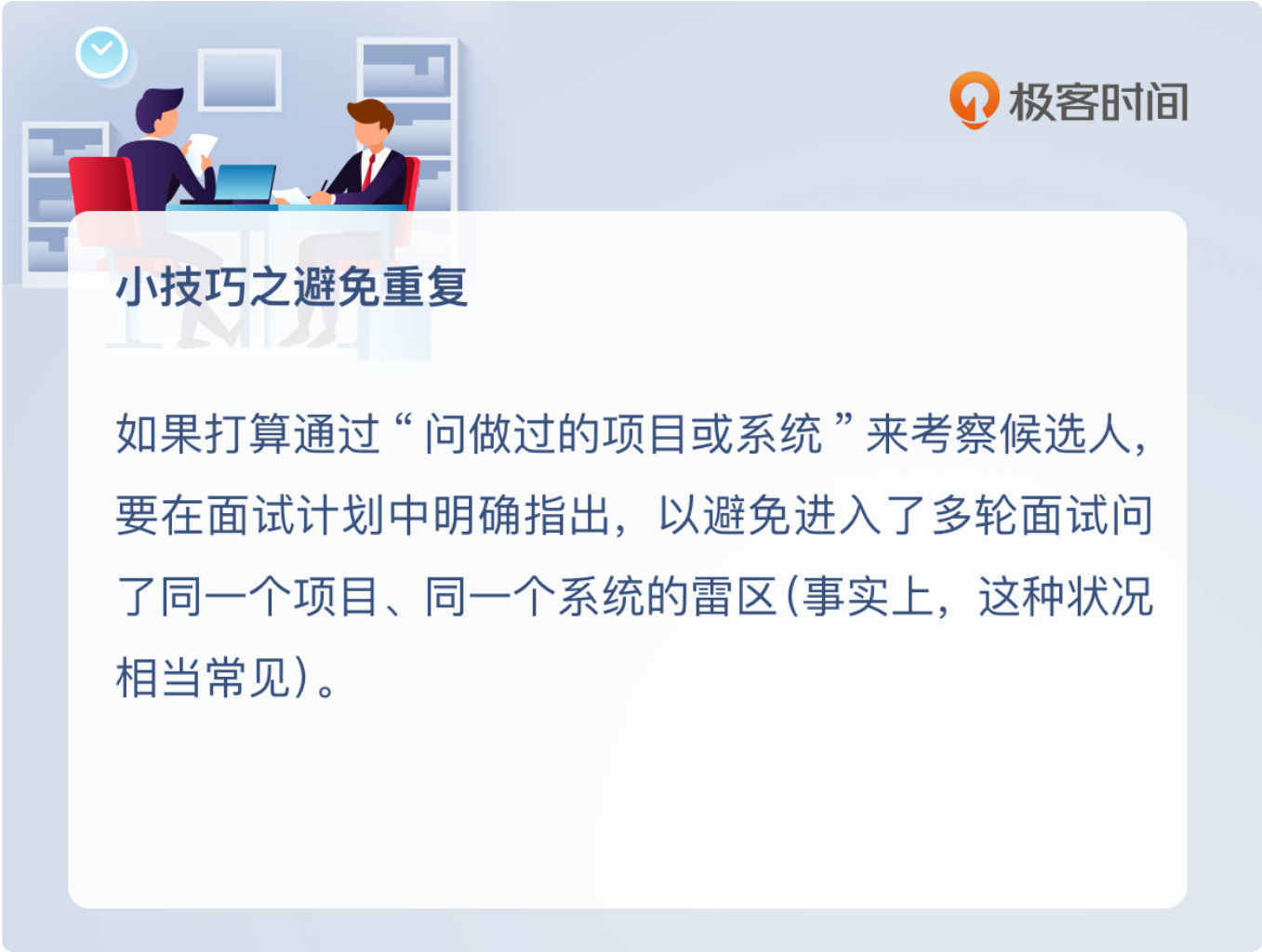
运用掌握的套路和方法来解决它。

事实上，在面试中，我也确实遇到个别候选人，虽然对于自己做过的项目说得头头是道，但给出一个新问题，却畏手畏脚，缺乏想法，表现不佳。

其次，必须指出的是，**这种方式，对于面试官的素质也有着相当高的要求。**

因为相对而言，这种方式下，谈论怎样的系统，包含着怎样的问题，涉及到怎样的技术，都是候选人所主导的，如果面试官并不具备丰富的经验，又恰好遇到一个不属于自己熟悉领域的软件系统，有时候就没办法提出一针见血的问题，也没法采集到充分且有说服力的考察数据。

在某些大厂的面试培训中，**培训老师甚至会明确建议，经验较少的面试官不要主导系统设计面试，而是主导相对容易操作的算法和数据结构的面试。**



对于发问，我们可以这样做：

> 你能否介绍一下，过去有哪一个你最熟悉，或是最引以为傲的软件系统？

我们也可以找到简历上最加以渲染的项目，而发问：

> 你在简历上说，你主导了一个半年期的项目 X，我对于这个系统很感兴趣，你能否介绍一下？

其实这两个方式的目的是一致的。

对于考察的过程，我觉得要从宏观和微观层面去分别把握。

在宏观上，观察候选人能否对他介绍的系统有着清晰的把握，有着整体的认识，而不是一下子就跳到某一个具体细节实现上去。比如说：

系统解决了什么问题？

系统的架构是怎样的？

模块、组件之间的交互是怎样的？

系统存在哪些问题，有哪些瓶颈？

如果再给一次重新设计系统的机会，哪些地方可以改善？

在微观上，对于候选人自己熟悉的、做过的部分，我们要选择几个点向下挖，并且挖掘得足够深，到“具体是怎样实现的”这样的程度，毕竟在现实中，夸夸其谈的人不在少数。

“深入”的核心，就是要抓住两个词——“具体”和“细节”，下面我通过一个小例子来说明。

比方说，候选人说，他最近主导了一个三个月期的性能优化项目，那我们就可以抓住这一点进一步细化。请看下面这个对话片段：

候选人：我最近在项目中把门户网站的性能优化了一倍。

面试官：我很感兴趣，能具体说说是什么方面的性能优化了一倍吗？

候选人：是网页的加载时间优化了一倍，以前平均一个页面要将近 3 秒，现在 1.5 秒就可以打开了。

面试官：是什么网页，门户网站的所有网页吗？

候选人：不是，是媒体详情页，不是媒体列表页。

面试官：哦，那你做了哪些改进才做到的呢？

候选人：我把详情页上的评论部分改成 Ajax 调用了，这样页面加载出来，用户浏览评论区时再去异步请求获取评论。

面试官：了解了，那你通过什么方法得知这个页面加载时间从 3 秒优化成 1.5 秒的呢？

候选人：我们做了性能测试，压测我们 beta 环境的系统，1.5 秒是 TP99 的值。

面试官：那在压测的过程中，系统的吞吐量是多少呢？

候选人：只有一个主请求，没有模拟静态资源访问的情况下，QPS 上限大概是 2000。

面试官：那在测试中你是怎样确认这个 QPS 已经达到上限了？

候选人：根据我们的单机压测模型，开始并发数逐渐增加，吞吐量线性增大，后来增长减缓，但到大概第 10 分钟的时候出现拐点，延迟突然增大，吞吐量也过了峰值走低。

这里抓住一个点深挖的目的，并不是考察候选人能否记得性能优化的每一处细节，而是要确认候选人是否真正参与了这个过程，是否对这个优化过程的几个核心问题有清晰准确的认识。

换言之，如果候选人重新做一个类似的性能优化项目，要看看他是否确实因为这个项目而具备了相当的功底。从中我们也可以看出候选人对于某项技术，到底把握到何种程度，对于做过的项目和系统，是否具备一定深度的理解。

需要说明的是，一个技术点的深挖要以拿到预期的考察数据为目标，可以问 1 分钟，可以问 10 分钟，这都是没有限制的。一旦拿到了预期考察的数据，不必过多纠缠，继续下一个环节。

当然，这个话题点一定要是候选人（至少是自称）熟悉的，并且，**深挖和关注细节不是纠缠于细节，我们不要把追问变成钻牛角尖，也不要把追问变成“考记忆力”。**

全新的问题

前面讲了我们怎样去考察候选人对于他做过的、熟悉的系统，到底有多少理解，下面我们再来讲讲，如果是一个新问题呢，候选人能否顺利地设计一个全新的系统来解决它？这两个方面其实是互相独立的。

那我们就用一个模拟案例来介绍，如果我们像第 4 讲说的，像做一个迷你项目一样，给出一个实际问题，层层递进，逐步细化、分解和延伸成一个系统设计的问题（如果你忘记了，请一定回看第 4 讲），我们又该注意哪些方面。那个具体问题就是：

> “请你设计一个网约车系统。”

并且我们也谈到了，经过细化以后，下面的系统设计考察，就是要求设计一个系统，来完成定位、叫车、搜索、接单等网约车系统的核心功能，如：

1. 乘客可以随时随地叫车；
2. 系统寻找邻近的司机并转发请求，如果无法接单，继续扩大寻找范围；
3. 司机可以抢单，即接受或者拒绝叫车请求；
4. 双方可以刷新自己在地图上的位置，也可以查看彼此当前的位置。

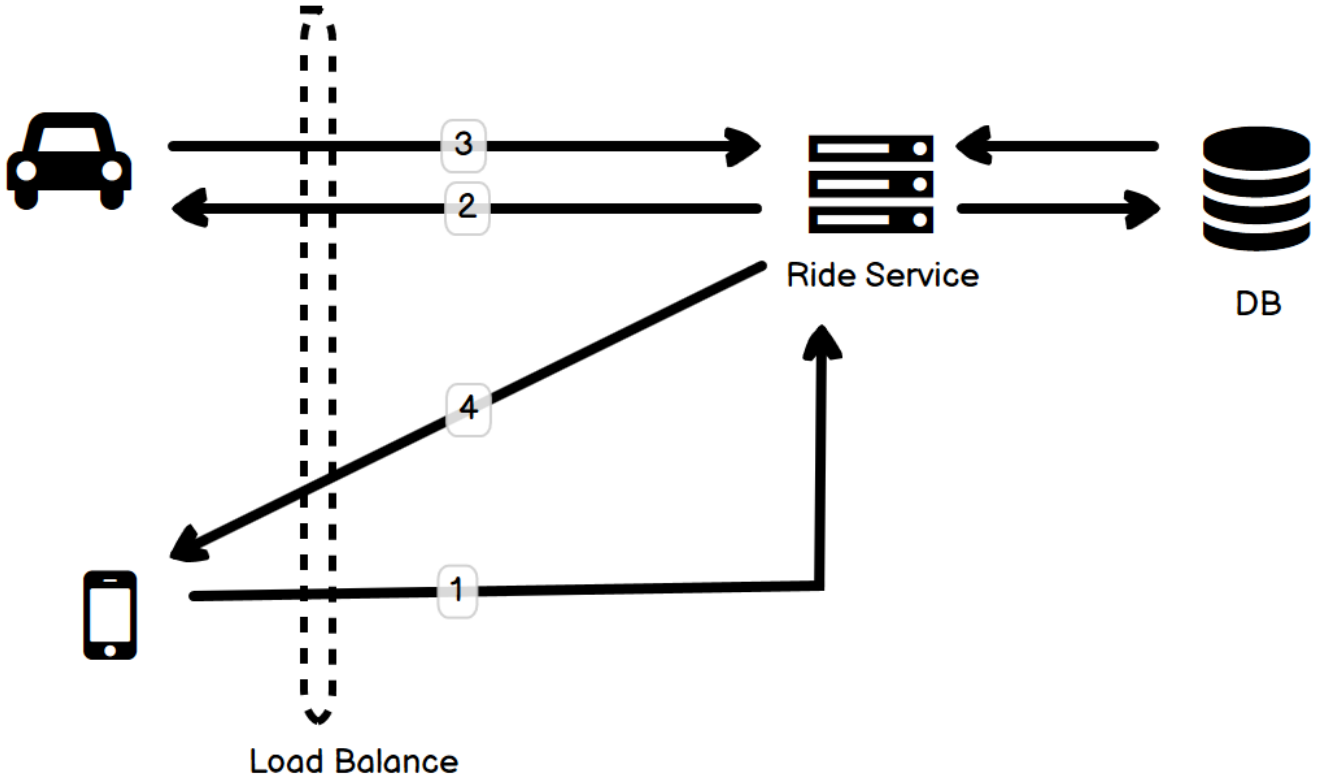
对于非功能需求，特别是系统容量的估算，我们后续将在专题中谈到；而对于功能需求的系统实现方面，我们继续来看后面候选人和面试官之间的对话片段。

第一阶段：客户端 - Service - 存储的基本层次建立

面试官：根据我们的讨论所确定的需求，这个系统中会有哪些主要模块？（要求澄清）

候选人：我觉得这个系统中，乘客可以叫车，司机可以接单，因此我打算创建一个 Ride Service 来接受乘客的请求，ride 的信息存放在数据库中。整个交互过程如下图：

1. 乘客请求乘坐，提交一个 ride（相当于一个请求乘坐的订单）；
2. Ride Service 将乘坐请求发给多个司机；
3. 司机抢单；
4. Ride Service 再将接单的信息告知乘客。



第二阶段：Service 解耦

面试官：嗯，那么其中的第 2 步和第 4 步，消息怎样从服务端传递给客户端呢？（要求澄清）

候选人：我觉得可以使用长连接，来保证消息能够第一时间送达，如果这一步失败，iOS 或者安卓平台还有统一的消息推送机制。

面试官：好，那么 Ride Service 怎么来寻找合适的司机来转发乘客的乘坐请求呢？（给出挑战）

候选人：我觉得 Ride Service 知道所有乘客和司机的位置，那就可以寻找最近的一些匹配，位置可以通过司机和乘客各自手机上的 app 来汇报，比如每十秒钟就汇报一次 GPS 位置。

面试官：可是我没有看到你画的图中有这个啊？（要求澄清）

候选人：哦，他们都汇报给 Ride Service。

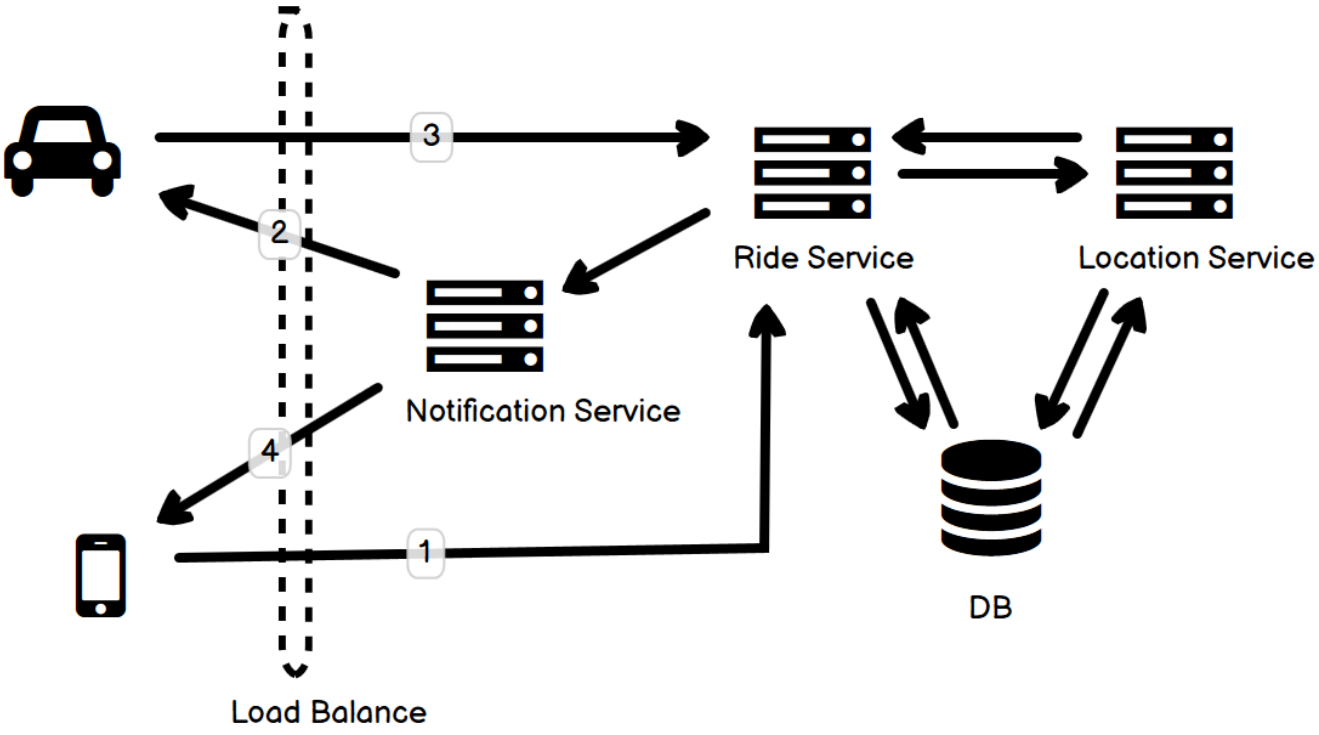
面试官：所以，位置信息归 Ride Service 管，乘车请求的匹配归 Ride Service 管，匹配上了以后，结果的通知也归 Ride Service 管，它一个 service 要管那么多事情吗？（给出提示）

候选人：嗯……我觉得可以解耦，根据单一职责的原则，让一个 service 只做一件主要的事情：

Ride Service 只管 ride 的匹配和决策；

Notification Service 负责通知消息；

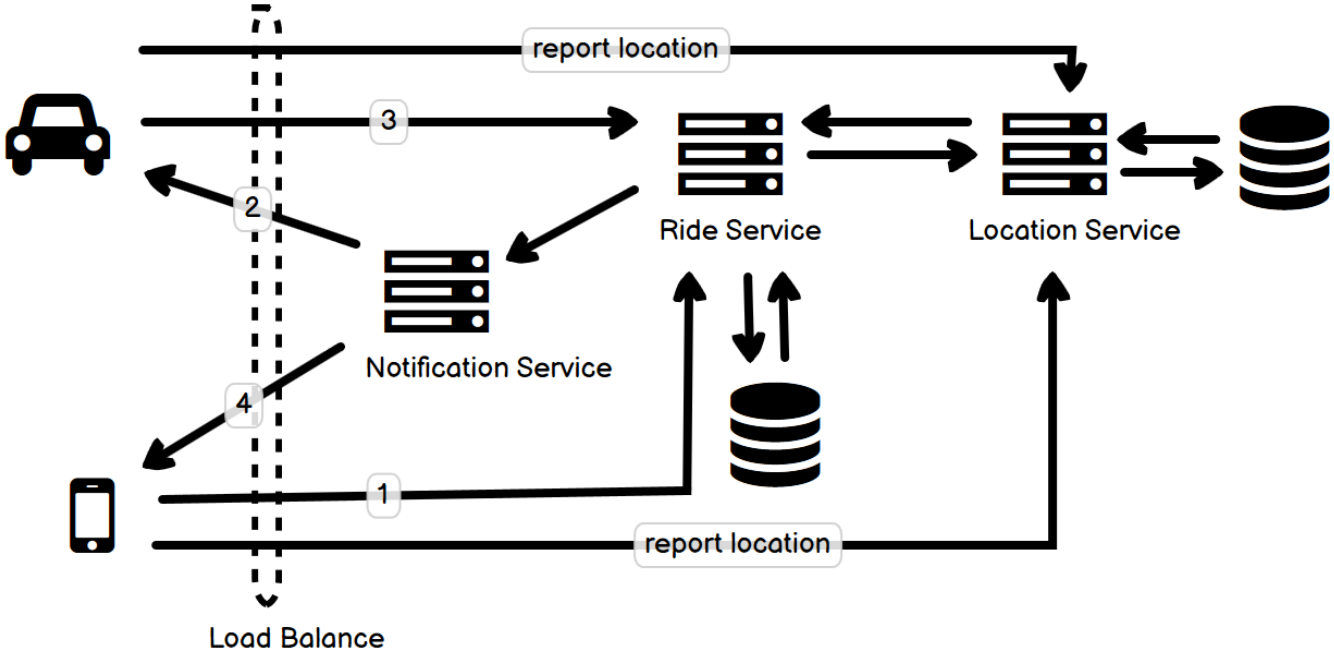
Location Service 负责维护司机和乘客的位置。



第三阶段：进一步解耦，补足缺失

面试官：嗯，确实看起来好多了。但是我还有个问题，司机和乘客汇报 GPS 位置的时候，从图上看，只有在乘客请求和司机接受 ride 的时候才会汇报位置信息吗？。（要求澄清）

候选人：哦，系统是需要实时获取位置信息，才好做匹配，我觉得 GPS 的信息也应该从 ride 的逻辑解耦开，应该有单独的位置汇报请求发给 Location Service。



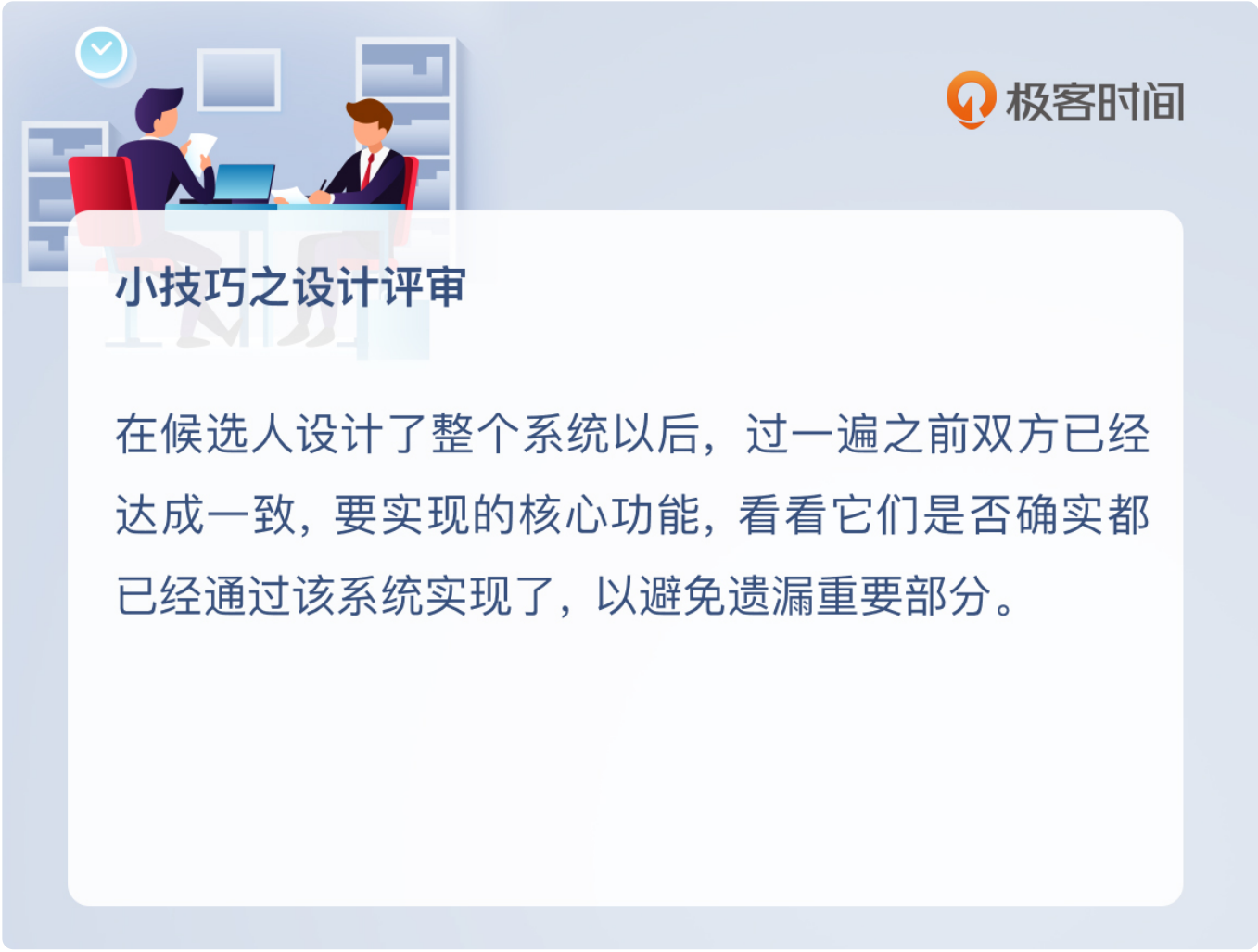
第四阶段：细化存储层设计

面试官：嗯，不错。我注意到 ride 数据和位置数据也随着 Ride Service 和 Location Service 的解耦而分离开了，这很好。那么，对于这两类数据的存储，你打算分别选用怎样的技术来实现呢？

候选人：ride 数据需要接受关系查询，位置信息需要根据司机、乘客的 ID 来查询，还需要能够快速找到邻近位置的司机和乘客……（对于这两类数据的存储，可以作为下一个话题来详细展开）

不知道你发现没有，这个小片段，就模拟了面试官使用“要求澄清”和“给出挑战”两大法宝，怎样领着候选人一步步攻克系统设计问题的过程，这个过程和上一讲的算法和数据结构面试，是类似的。

其中，对于位置数据的存储，如果感兴趣的话《全栈工程师修炼指南》的第 26 讲中，我做了详细的介绍，你可以做延伸阅读。



总结与思考

好，今天的内容主要就是这些。

我先给你举了一些系统设计面试的踩坑案例，然后又给你分享了两类主要的面试方式，深挖候选人做过的系统，或者和他一起讨论一个全新迷你项目的实现，它们都能考察候选人对于系统的理解，但是又各有侧重。

你可以结合我为你还原的面试现场对话，仔细体会系统面试的操作方法。

这里我需要额外强调的是，系统设计就和算法一样，我们还是更应该关注基础，选择合适的问题，可以问，但不要一味追求高大上的“海量存储”、“大数据”这样的问题，这和算法和数据结构的面试不要过度追求“偏题”、“怪题”是一样的。

毕竟，能将生活中那些实际的问题，选择合适的工程技术来解决，才是软件工程师最根本的使命。

极客时间

在最后，我想请你聊一聊你在阅读后的想法，特别是你是否有系统面试的经历，你觉得有哪些特别重要的方面，能否在留言区谈一谈呢？我会积极回复你的想法和问题。

好，我是四火，我们下一讲见！

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 06 | 算法和数据结构考察：怎样有层次地驾驭算法考察？

下一篇 08 | 其它技能考察：见微知著，不可忽略的其它考察点

精选留言 (2)

写留言

qinsi

2021-04-26

所以有时候你觉得面试官没有问你很深入的问题，整个过程都在谈笑风生，结果最后还是没有被录取，可能只是因为面试官觉得你基础不行不需要深挖，给大家都留点面子。

展开

3

fakefish

2021-04-24

关于候选人思路不一致，但遇到按照候选人第一次思路下去会在十几分钟写不完，我会让他在一开始提示换一个思路，免得后面时间不够用

展开

作者回复: 具体是写什么写不完呢?

2

1